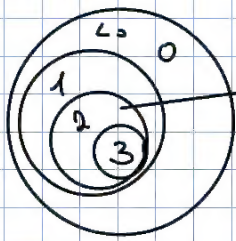


Gerarchie di Chomsky



$L = \{a^n b^n \mid n \geq 1\}$ può essere generato da grammatiche di tipo 0 purché ometta ϵ -produzioni sia da grammatiche di tipo 2 (non contestuali) non tipo 3

L_0 è tipo 0 ma potrebbe non appartenere a tipo 1/2/3

Osservazione:

Se analizziamo grammatiche di tipo 2/3 ci consentite la presenza di una ^{più} ϵ -produzione, mentre nel caso di tipo 1 è omessa una ϵ -produzione solo nell'unione in S , e condizione che questo non compaia mai nelle parte destre di una produzione

esempio: "poniamo
problema senza fermarci"

Grammatica Tipo 3?

$G: S \rightarrow aS \mid bA$

$A \rightarrow bA \mid \epsilon$

Togliamo le ϵ -produzioni per fare esercizio

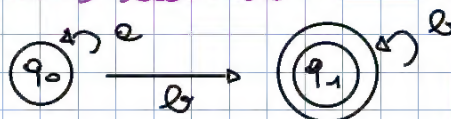
$G: S \rightarrow aS \mid bA$

$A \rightarrow bA$

relative espressione regolare

$a^* b b^*$ oppure $a^* b^+$

ASFD relativo



CAPitolo 4 : Linguaggi non contestuali

Quindi, in sintesi, queste sono state introdotte come strumento per definire le strutture del linguaggio naturale.

Possiamo caratterizzare le basi di un linguaggio con soggetto e predicato e quindi comporre una frase

$F \rightarrow SP$ frase \rightarrow soggetto + predicato

$P \rightarrow V \mid VC$ predicato \rightarrow verbo \mid verbo + complemento oggetto

Queste strutture sono sufficientemente semplici e quindi in alcuni contesti inadeguate come strumento per il linguaggio modulare.

2) **Lingueggi**: gemeroti non contestuali rappresentano un sottoinsieme proprio dei lingueggi

3 linguaggi di programmazione rappresentano un sottosistema proprio dei linguaggi non contestuali e li consideriamo come alberi.

Alberi di derivazione o alberi sintattici

$G: S \rightarrow e S G A \mid eb$

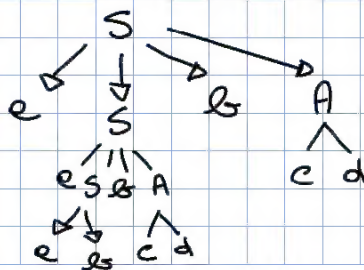
$$A \rightarrow cAdled$$
$$\rightarrow aSbe \rightarrow eebbA \rightarrow eebbcd \in \mathcal{L}(G)$$

La grammatica introduce il linguaggio $\mathcal{L}(G)$ e $\mathcal{L}(G) \in \mathcal{L}(G)$

un esempio di albero di

derivazione di una

stränge $\in \mathcal{L}(G_1)$



Queste precomuniche usano una memoria gestita a pile
e nel corso di linguaggi di programmazione queste corrispondono
ad un'analisi sintattica "parsing" relative al programma di
"tradurre" le stringhe in un'altra stringa più elementare,
capibile dal PC

CAPITOLO 5:

Macchine di Turing e calcolabilità secondo Turing

Le mt sono il modello di calcolo di riferimento fondamentale
sia nell'ambito delle teorie di calcolabilità sia in quella
delle teorie della complessità computazionale

L'obiettivo è quello di formalizzare il concetto di calcolo
allo scopo di stabilire l'esistenza di metodi algoritmici
per il riconoscimento dei termini nell'ambito dell'aritmetica

- Elevata semplicità strutturale
- Potere computazionale

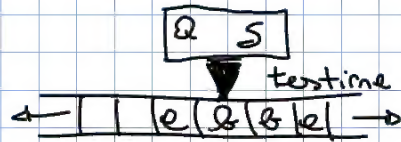
Le macchine di Turing viene definite come un dispositivo
operante su stringhe contenute su una memoria esterna
(nastro) mediante passi elementari definiti da fun.
di transizione

Macchine di Turing e nostro simbolo

Nelle MT vi è un nastro potenzialmente infinito diviso in celle contenenti ciascuna un simbolo appartenente ad un dato alfabeto Γ ampliato con il carattere speciale \bar{b} (blank) che ^{→ forma} rappresenta la situazione di una cella non contenente caratteri.

Le MT lavorano su un nastro, o vi è una testina che scorre su di esso in entrambe le direzioni.

Si può leggere o scrivere caratteri dell'alfabeto Γ oppure \bar{b}



Ad ogni istante la MT si trova in uno stato appartenente ad un insieme finito Q e il meccanismo che fa evolvere la computazione delle macchine è la funzione di transizione

δ parte da uno stato e da un carattere \rightarrow stato

Definizione MT

Una macchina di Turing deterministica (MTD) è una sextupla $M = \langle \Gamma, \bar{b}, Q, q_0, F, \delta \rangle$

Γ = alfabeto ^{" \bar{b} rappresentato"}

\bar{b} = blank $\notin \Gamma$

Q = insieme degli stati

q_0 = stato iniziale

F = stati finali $F \subseteq Q$

δ = funzione di transizione definita così:

$$\delta: \underbrace{(Q - F)}_{\text{STATO NON FINALE}} \times \underbrace{(\Gamma \cup \{\bar{b}\})}_{\text{SIMBOLO}} \mapsto \underbrace{Q \cup (\Gamma \cup \{\bar{b}\})}_{\text{stato simbolo}} \times \underbrace{\{d, r, i\}}_{\text{spostamento}}$$

$\{d, \rightarrow, i\}$ indicano lo spostamento e dx, \rightarrow, i o ensemble di spostamento delle testine

$\bar{F} = \Gamma \cup \{\bar{q}\}$ per compattezza
e soprattutto $\Sigma \in \bar{F}$

Esempio:

$$\delta: (Q-F) \times (\Gamma \cup \{\bar{q}\}) \rightarrow Q \times (\Gamma \cup \{\bar{q}\}) \times \{d, \rightarrow, i\}$$



$$(q_3, b) \mapsto (q_4, e, d)$$

$$\delta(q_3, b) = (q_4, e, d)$$

Le macchine usate per accettare stringhe vengono dette accettatori, mentre quelle usate per calcolare funzioni vengono dette di tipo trasduttore

Funzione caratteristica del linguaggio: che è definita da Σ^* e $\{0, 1\}$ ed è ovviamente una funzione che assume valore 1 per ogni stringa del linguaggio o valore 0 altrimenti.

Forme normali di Greibach

$$A \rightarrow \alpha e \quad A \in V_N \quad e \in V_+ \\ \alpha \in V_N^*$$